Objectgeoriënteerd Programmeren: WPO 2

1. Inhoud

Klassen, objecten, methoden, properties, private vs. public, velden, instantie, reference to object, this.

2. Oefeningen

- Demo 1: Bugs
- Demo 2: Kleurcodes
- A: Klasse Punt
- A: Biljarttafel
- A: Rekenmachine
- A: Converter
- A: Softwarelibrary
- E: Regendruppels
- X: Snake
- X: Gravity

2.1 Demo 1: Bugs

Schrijf een programma die een populatie vliegen simuleert waarin de vliegen naar een hoopje suiker aangetrokken worden.

1. Schrijf een klasse "vlieg" waarin je de positie (x,y) en de kleur van de vlieg bijhoudt. Alle vliegen worden in een lijst bijgehouden. Zorg ervoor dat je deze eigenschappen van buiten de objecten ook kan opvragen en wijzigen (getters en setters en/of properties). Voordat de vliegen kunnen bewegen worden deze eerst getekend. Maak hierbij 25 vliegen aan die je op een random positie op de canvas zet (grootte = 300 bij 300 pixels). Gebruik hiervoor een constructor die de 3 eigenschappen als argument opneemt.

2. Omdat we niet willen dat vliegen buiten ons scherm komen, voorzien we ook een *check*-methode om na te gaan of de vlieg wel degelijk op het scherm is. Indien niet wordt de vlieg op positie (300,300) geplaatst. Deze check verloopt in een aparte methode die *private* is.

3. Eens dit gelukt is voorzie je de vlieg van een nieuwe methode (public) die de vlieg toelaat te bewegen naar het midden van het scherm (hoopje suiker, grijs vierkantje van 25 bij 25 pixels). Deze methode genereert randomgetallen om nieuwe x- en y-waarden te berekenen en kijkt na of de vlieg hiermee dichter bij de suiker komt. De randomgetallen liggen tussen -5 en +5 en worden hierbij aan de huidige coördinaten opgeteld. De vlieg wordt verplaatst indien de nieuwe x- en y-waarden dichterbij gelegen zijn.

Het geheel wordt getekend in een canvas van 300 bij 300 pixels. Gebruik een timer die alle vliegen in een loop tekent en ervoor zorgt dat de vliegen van positie kunnen veranderen.

2.2 Demo 2: Kleurcodes

Schrijf een programma die het mogelijk maakt om de kleurcodes van weerstanden uit te rekenen. Volgende functies worden voorzien:

- een functie om een enkele kleur (string) om te zetten in een getal,
- een functie om een getal van 0 t.e.m. 9 om te zetten in een kleur (string),
- een functie om 3 opeenvolgende kleuren (array van string) om te zetten in een weerstandswaarde en,
- een functie om een weerstandswaarde om te zetten in 3 kleuren (array van string)

Maak bij de implementatie van dit programma gebruik van een klasse met static methoden.

2.3 A: Klasse Punt

Deze opgave wordt in verschillende tussenstappen behandeld.

1. Schrijf een klasse "Punt" waarin je de velden X en Y definieerd. De X en Y velden zijn beide kommagetallen. Zorg ervoor dat je deze velden kan aanpassen via gepaste methoden en/of properties. Deze zijn uiteraard *public*.
2. Schrijf een methode (clone) die toelaat om een kloon van het huidige object te maken. Deze methode retourneert een waarde van het type "Punt". In deze methode kopieer je alle eigenschappen van het moederobject in het nieuwe object.
3. Schrijf een methode (add) die beide velden kan ophogen met waarden die als argument meegegeven worden. Het resultaat hou je bij in het huidige object.
4. Schrijf een methode (addPoint) die als argument een Punt opneemt en de velden optelt met de velden van het huidige object. Bv. als het ene object (3,7) en het 2de object (-5,7) bevat, dan wordt het resultaat (-2,0) in het huidige object.

Voorzie voor elk van deze operaties een gepaste knop. Je mag 2 tot 3 objecten bijhouden in het programma. Het resultaat van elke operatie print je af in een label. Voor de eenvoud van het programma mogen de getallen handmatig geprogrammeerd worden.

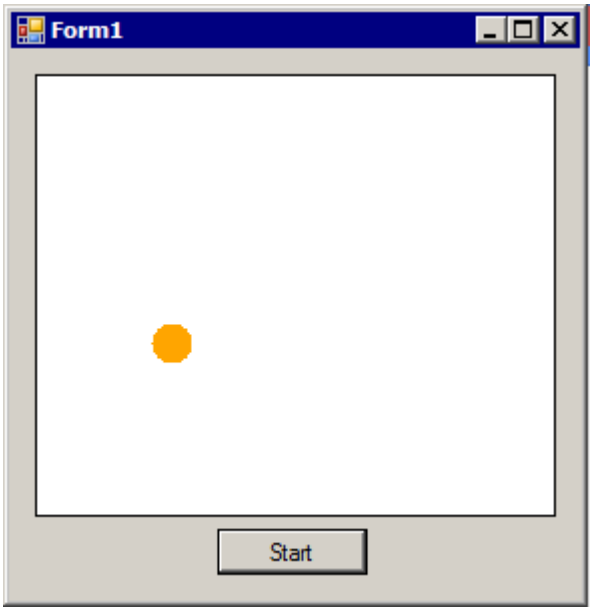
2.4 A: Biljarttafel

Schrijf een programma dat een biljarttafel kan simuleren. Maak een klasse 'Bal' aan waarin de nodige eigenschappen van elke biljartbal opgenomen worden. Voorzie nodige getters en setters om de eigenschappen van de biljartbal op een veilige manier te kunnen aanpassen. Beperkingen en features die hierin opgenomen dienen te worden:

- een biljartbal kan niet buiten het veld komen. De X en Y waarden hebben een boven- en onderlimiet.
- elk biljartbal heeft een snelheid in de X en Y richting (snelheidsvector)
- een biljartbal heeft natuurlijk ook een kleur.

Voorzie methoden om de biljartbal te kunnen updaten. Het uitrekenen van de nieuwe positie en snelheidsvector gebeurt binnen de klasse 'Bal' zelf.

Een voorbeeld van deze applicatie kan teruggevonden worden in figuur 1.

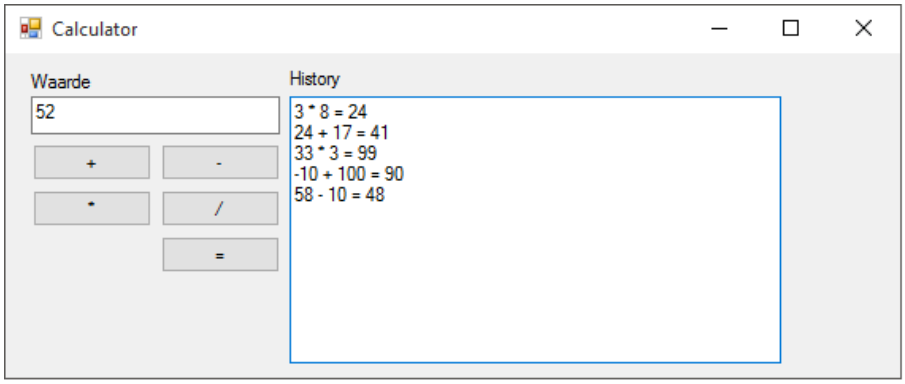


Figuur 1: Voorbeeld biljarttafel

2.5 A: Rekenmachine

Matlab en andere gelijkwaardige wiskundige softwaretools laten toe om wiskundige vergelijkingen uit te rekenen. In deze opgave zullen we een gelijkaardig programma schrijven dat de gebruiker toelaat om eenvoudige vergelijkingen bestaande uit 2 getallen en een operator uit te rekenen. Laat de gebruiker toe om achtereenvolgens een getal, een bewerking en een getal in te geven. Houd de 2 getallen (en het resultaat) en de operator in een object bij en reken de uitkomst uit.

Matlab houdt een geschiedenis bij van de voorgaande ingevoerde bewerkingen. Hou dus net zoals Matlab de bewerkingen bij in een geschiedenisvenster (zie figuur 2).



Figuur 2: Voorbeeld van het rekenmachine

2.6 A: Hexconverter

Schrijf een programma dat toelaat om omzettingen tussen verschillende getalformaten uit te voeren. De bewerkingen die gevraagd worden zijn:

- conversie van hexadecimaal formaat (string) naar integer,
- conversie van integer naar hexadecimaal (string) formaat,
- conversie van integer naar binair (string) formaat,
- conversie van binair (string) formaat naar integer,
- conversie van integer naar octaal (string) formaat,
- conversie van octaal (string) naar integer formaat,
- conversie van binair naar hexadecimaal en omgekeerd,
- conversie van octaal naar hexadecimaal en omgekeerd,
- conversie van octaal naar binair en omgekeerd.

Maak gebruik van een conversieklasse met static methoden.

2.7 A: Softwarelibrary

Software wordt typisch bijgehouden in softwarebibliotheken. Hierbij worden per stuk software de volgende eigenschappen bijgehouden.

- de programmeur (of naam van de groep),
- de naam van de software,
- het versienummer van de software (integer).
- de datum van uitgave (dag, maand en jaar)

Schrijf een applicatie die toelaat om de referenties van de stukken software bij te houden. Voorzie hiervoor een lijst die objecten van de klasse 'Software' bijhoudt. Deze klasse bevat minimaal bovenvermelde eigenschappen tezamen met de nodige getters en setters. Volgende beperkingen en features worden voorzien in deze applicatie.

- Elk stuk software komt maximaal 1 keer voor. Hiervoor wordt via een nieuwe methode in de klasse de naam van de software vergeleken. Schrijf hiervoor de methode *check* die true retourneert indien de naam overeenkomt met de naam van het huidige object (en false anders). Bij true wordt de recentste versie bijgehouden (hoogste versienummer).

- Indien een stuk software 2 maal zou voorkomen maar geschreven is door verschillende programmeurs, wordt gevraagd welke versie bijgehouden moet worden. Ook deze vergelijking wordt binnen de klasse zelf uitgevoerd.

- De lijst met stukken software wordt in een listbox weergegeven.

De gebruiker kan stukken software toevoegen en verwijderen.

2.8 E: Regendruppels

In een weerkundig model worden vallende regendruppels gesimuleerd. Als programmeur wordt jou de opdracht gegeven om hiervoor een 2D model op te stellen. In dit model worden 50 regendruppels voorzien, elk met een diameter van 20 pixels. Elke regendruppel bevat volgende eigenschappen:

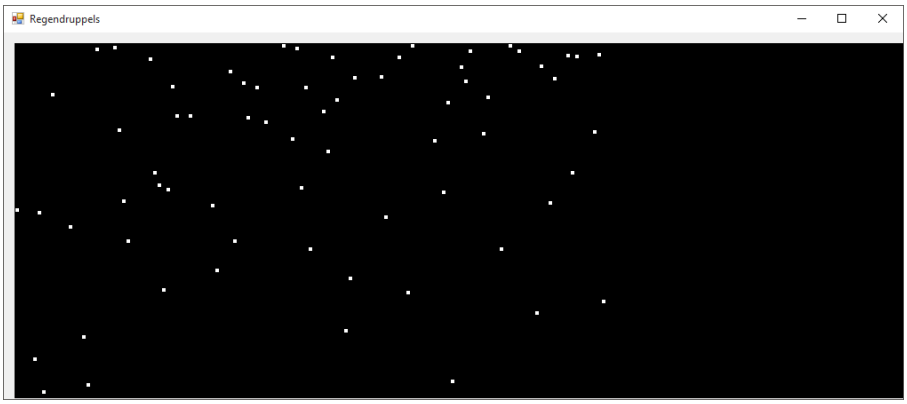
- de hoogte t.o.v. de grond (y),

- de snelheid waarmee de druppel naar beneden valt. Deze start met 0, en neemt toe met de valversnelling (g=9.81),

- de horizontale positie (x), deze wordt eenmalig ingesteld en verandert nooit,

- een eigenschap of deze druppel al dan niet actief is.

Door een randomgenerator te gebruiken, kan men beslissen of een inactieve druppel getekend zal worden. Zolang een druppel niet getekend wordt (inactief), blijven de hoogte en de snelheid van de druppel op 0. Initialiseer deze waarden van zodra de druppel getekend mag worden. De druppel wordt dan ook als actief gezet. Met de randomgenerator kan dus enkel beslist worden of een druppel van inactief naar actief overgaat. Zolang de druppel getekend wordt (kijk na wanneer de druppel de grond raakt), wordt de druppel als actief beschouwd. Wanneer deze grond raakt wordt deze inactief. Gebruik een array van 50 druppels, en hanteer een minimale vensterbreedte van 1000 pixels (50 druppels bij 20 pixels per druppel).



Figuur 3: Voorbeeldprogramma van de regendruppels. De regendruppels zijn hier wit.

2.9 X: Snake

Snake is het spel waarbij je een slang bestuurt en de slang objecten moet "opeten". Telkens de slang iets opeet wordt deze langer.

- De objecten die de slang moet opeten komen op random plaatsen op de panel. Hiervoor genereer je een random x en y coördinaat en teken je het object op die plaats. Wanneer de slang het object opeet wordt een nieuw object weergegeven.

- De slang wordt voorgesteld door vierkantjes. Elke keer de slang iets opeet, wordt een vierkantje toegevoegd. Schrijf hiervoor een klasse (vb: slangsegment) met velden Xpos en Ypos en voorzie hiervoor de nodige getters en setters. Maak een object aan telkens de slang langer wordt. Hou alle objecten bij in een lijst.

- Voor de beweging van de slang wordt telkens één nieuw coördinaat berekend (enkel voor het eerste segment, afhankelijk van de richting). De rest van de coördinaten wordt gewoon doorgeschoven naar het volgende segment.

Hint: De slang bestaat uit blokjes die als vierkantjes kunnen worden voorgesteld. De X en Y coördinaat van elk blokje mag gelijkgesteld worden aan de linkerbovenhoek van dat vierkantje. Verdeel de panel (coördinaten) in stukken die evengroot zijn als de grootte van één blokje van de slang.
Hint: Maak van deze zijdelengte ook gebruik om elk op te eten voorwerp op een veelvoud hiervan uit te zetten. Detecteren wanneer de slang het voorwerp kan opeten, wordt dan gedaan door zowel de X en Y coördinaten van de kop van de slang te vergelijken met de X en Y coördinaten van het voorwerp. Indien deze overeenkomen, wordt het voorwerp opgegeten en neemt de slang met 1 vierkantje toe.
Hint: Gebruik een enum om de richting van de slang bij te houden. Telkens een richtingspijl ingedrukt wordt, wordt deze enum aangepast. Gebruik deze enum om de positie van het voorste

segment aan te passen.
Hint: Maak van de slang ook een object.

2.10 X: Gravity

In ons zonnestelsel bevinden zich een zeer groot aantal hemellichamen waaronder onze vertrouwde zon en enkele kometen. Afgezien van een aantal niet gemeenschappelijke eigenschappen, worden hemellichamen gekenmerkt door de massa, de grootte (diameter), de bewegingssnelheid, de kleur en de positie (2D). Maak een klasse aan dat deze eigenschappen van de hemellichaam bevat. In ons zonnestelsel worden alleen de zon en één komeet getekend. Teken beide hemellichamen door deze de gepaste eigenschappen te geven.
Natuurlijk blijven hemellichamen niet stationair binnen het zonnestelsel evolueren. Deze bewegen. De beweging van de hemellichamen wordt voornamelijk bepaald door de huidige snelheid van het hemellichaam en de zwaartekracht dat deze ondervindt t.g.v. andere hemellichamen. De grootte van de zwaartekracht t.g.v. een ander hemellichaam wordt beschreven als:

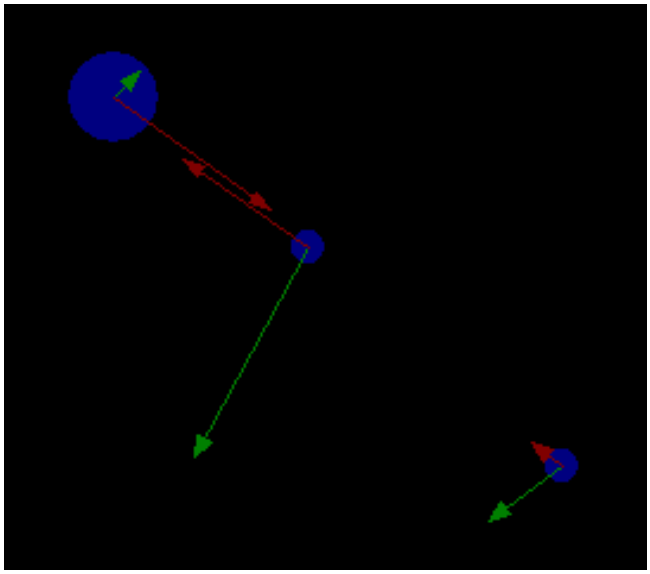
$$F = G \cdot \frac{m_1 \cdot m_2}{r^2} \tag{1}$$

waarbij F de grootte van de zwaartekracht is, m_1 de massa van hemellichaam 1 is, m_2 de massa van hemellichaam 2 en r de afstand tussen beide hemellichamen. Ga dus na hoe de afstand tussen beide hemellichamen berekend kan worden (maak hiervoor een functie aan).
In deze opgave mag de zon statisch blijven, dus enkel de positie van de komeet verandert. De nieuwe positie en snelheid van de komeet worden als volgt bepaald:

$$p_{komeet} = p_{komeet} + v_{komeet} \cdot t + \frac{at^2}{2} \tag{2}$$

$$v_{komeet} = v_{komeet} + at \tag{3}$$

De zwaartekracht, snelheid en positie worden uitgedrukt in vectoren (dus X en Y). Het verband tussen het vectorieel verband en de numerieke waarden in formules 1, 2 en 3 wordt gegeven door de hoek tussen de posities van de zon en de komeet.
Gebruik een timer om het effect van de gravitatie op de komeet weer te geven. Teken ook de situatie.
Hint: $F = m \cdot a$



Figuur 4: Voorbeeld van 3 hemellichamen. Hier worden ook de snelheidsvector (groen), samen met de zwaartekracht (rood) dat ondervonden wordt door elk lichaam getekend. De pijlen hoeven niet getekend te worden.